An article in a British magazine

The ten commands: will they change your life?

There's no need to step through fire to bring the tablets down from Sinai. John James takes a sceptical look at ten (or is it nine?) new add-on machine-code subroutines for the PET, holds them nervously in the palm of his hand, and concludes they may change your life, too.

WHAT EXACTLY is the Basic Programmer's Toolkit? Well, answering physically, it'll depend on whether you have an 'old' or 'new' PET. And how do you tell? If you have one of the small keyboards, then for our purposes you have an 'old' PET; a big keyboard (and therefore a separate tape cassette unit), and you have a 'new' PET.

The only effect this has on the Toolkit is on the type of Toolkit you get.

For an 'old' PET, you get a small printed circuit board, with a chip and some other bits mounted on it, and a connector loose-wired to its edge. The board has another connector mounted directly on it, which lets it be plugged straight into the memory expansion board port on the right-hand side of your PET. The loose-wired connector is then plugged into the second cassette port, just round the corner, at the back.

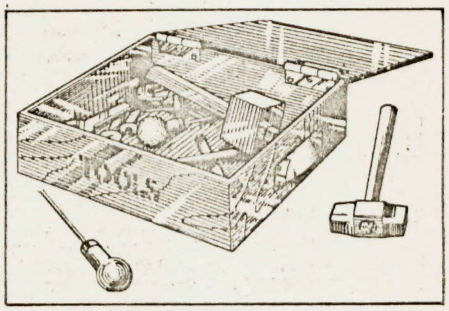
For a 'new' PET, you simply get a chip. This you take in nervous fingers (try holding a chip that cost over sixty quid, and see if you aren't nervous), to plug straight into an empty socket on the main board inside your PET.

Obviously, installation on an 'old' PET couldn't be easier, so there's no point in covering the question 'can you do it simply?'. Fitting a 'new' PET sounds a bit more tricky, so how hard is it?

Nothing could be more straightforward, thanks to one of the best instruction books I've seen in many a day. If you follow the book, you really can't go wrong. It's worth saying here, for the knowledgeable, that the book also covers all the things you'd expect, like avoiding static discharge, bent pins, and so on. In other words, fitting should be no problem at all — except if you've had expansion memory fitted to your PET. Here, the book recommends that you 'contact your dealer for installation information'.

Now there is room for criticism here, since you won't be reading the book until after you've got your Toolkit. Have you ever bought a super-gadget-gizmo, hurried home in high excitement, torn the wrapper off, and then found you hadn't got the necessary battery because nobody said you'd need one? Familiar with the frustration this causes?

Then be warned. If you have got expansion memory, tell your dealer at the time of buying, and make sure that he can advise you.



Once you've fitted the Basic Programmer's Toolkit, what will it do? Here, I'd like to come back briefly to the instruction book that comes with the Toolkit. It's 34 pages long, nicely printed, totally explicit, clear as crystal, loaded with example programs, and best of all, originally written by Gregory Yob.

The cognoscenti will have twitched at that magic name; others should know that Gregory has not only been in personal computing since Heaven knows when — to many, he *is* personal computing.

This means that you'll get the best out of the Toolkit right from the word 'go', and the best can meant quite a lot. Yet there is room for another criticism. The Toolkit is widely advertised as adding '10 powerful new commands to PET's Basic', which I think is not entirely fair.

In fact, you get nine new commands (we'll have a look at how powerful they are in a moment); the tenth isn't really a command, in my view, since it simply switches off a couple of the other nine.

So what are these nine powerful commands? The best way of answering that is to look at them one by one, but before we do that, let's return to that magic moment just after you've fitted Toolkit, and you're about to start trying it out.

You have to turn it on, and this you do very simply by entering SYS 45056. PET should respond immediately by displaying (C) 1979 PAICS.

From that point on, all nine active commands are available until you switch PET off. You get them back, after powering down, by entering the SYS instruction again.

Reviewing the nine commands gives me a choice: I can tell you about them alphabetically, or in order of personal favouritism, and I'm going to opt for the latter.

Let's start with RENUMBER.

Now this one is really good (not that the others aren't, but I said I'd start with my favourite!). The major point to make immediately is that RENUMBER is fully professional.

In other words, forget about those previous methods, using add-on subroutines, which required you to go right through the program, looking for line numbers, to make sure each one had enough space before it for whatever it might become when renumbered.

The Toolkit RENUMBER expands or contracts line numbers, and closes everything else on the program line up tight to the new number, whatever it might turn out to be.

You activate RENUMBER by typing the word and entering it. If you do this, Toolkit assumes you want to start at Line 100 and go in steps of 10.

If you want to start at any other line number, and/or go in other increments, you simply say so and it happens — and pretty fast too. I linked four programs together, all about 6K long, and renumbered the lot in less than 15 seconds.

Of course, it depends on the number of renumbers (if you get my meaning), but you shouldn't have any complaints about the time you hang about.

There are two other goodies in RENUMBER.

First, if renumbering will take your line numbers over the maximum permitted, the routine aborts, and no renumbering takes place. You'll like that if you've ever 'crashed' with other methods!

Second, if the routine comes across a line number that doesn't exist later in the program (as in GOTO X, and X isn't there, for instance) then it quite deliberately renumbers that to 63999.

This ties in nicely with my next most-liked command, FIND.

FIND locates all lines that contain absolutely anything you like to specify. It's important to be clear about this, because the advertising for Toolkit isn't. The advertisements say that FIND locates

lines 'containing a desired character string'. That's quite true, but it's misleading, in that you might well think that all you can ask FIND to do is look for strings.

Not so. FIND will do much more than that (which certainly means the advertisements err on the right side!). FIND will find anything. If you want to know which lines have the variable 'G' in them, and you enter FING G, then every line with the variable in it will be found and displayed.

Note that FIND G won't find GOTO or GOSUB, just because they have a 'G' in them, which saves a lot of confusion. To find GOTO statements, you need to enter FIND GOTO, and the same holds good for any other Basic statement. If, on the other hand, you enter FIND "G", then every quoted string that contains the letter G, whether it's on its own or contained in a word, will be found and displayed, which is not only useful, but fun.

Now you'll see the point of renumbering non-existent lines to 63999. All you need to do to find them is enter FIND 63999, and presto! there they all are, displayed for your pleasure. And, what's more, you can then go right ahead and edit them on screen, which helps resolve little local difficulties quite quickly.

Keeps on adding

I think my third favourite must be APPEND. This one adds program to program, for just as long as you have memory available to keep doing it. You only have to keep one thing in mind: APPEND does not replace program lines in memory with new similarly-numbered lines from the material you're appending, not does it interleave program lines. Quite simply, it does exactly what it says: it keeps on adding program lines to the end of whatever might already be in memory.

It does this from tape only, however, and will not append from disk or other devices. I don't see this as a very great disadvantage, but perhaps some people will.

All I can say is that I've already found the command incredibly useful, in only a few days of playing about. If I have a favourite subroutine or whatever on disk, I don't really feel it's a great hardship to off-load on to cassette so that I can use the APPEND facility.

My fourth favourite? Well, we're getting to the photo-finish stage now, but maybe a whisker or so in front is HELP. You use this when that lovely point comes, as you finish typing the world's finest program, which of course is totally error-free, and run it, when everything judders to a standstill halfway through, with PET beadily blinking ERROR at you.

You look at the line allegedly in error, and it seems perfect, just like the rest of the program.

Fret not; if you'd like to know where that little but is lurking, enter HELP. Up comes the line again, but this time a bit of it will be in reverse field. This, says the manual, is where the error is. Well, I found this wasn't strictly true, but it was close enough to make no difference. Sometimes the reverse field is to the left or right of the error, sometimes a character or so distant, but it's never far away. After all, if the error was something missing, it'd be hard to put a reverse field on what wasn't there!

Mounting panic

The fifth and sixth commands are similar: TRACE and STEP. Both put a little reverse-field window on the top right-hand corner of the screen, which displays up to six line numbers. With TRACE, the program runs steadily, and the line numbers being executed whip up the litttle window at a high rate of knots.

The manual says that TRACE slows the running of a program considerably, and that pressing the shift key slows it still more, to around two lines per second. Even at that, though, it's still moving fast, and I haven't so far found TRACE overpoweringly useful. The only effect it's had on me to date is to induce a mounting panic as I tried to keep up with what was happening. These are early days, though.

STEP seems rather more sensible, but to be harsh, it's merely an extension of TRACE, in that it displays one line number only, and that's the one that's just been executed.

For debugging, my vote goes to STEP, however. Not that my programs ever have bugs in them, of course, but I can always offer my help to those less fortunate!

One smart thing about both TRACE and STEP: the reverse-field window overrides any screen display, which avoids peering at line numbers through overwritten screen characters. This is a disadvantage too though, because it's possible (but, to be fair, odds-on unlikely) that a bug you're trying to spot will be right under that little window!

DUMP comes next, and I have a feeling that I'm going to love it more and more. You use it most often at the end of a program, and entering it causes every single variable (excepting arrays) to be printed, with the value each has at that time.

Now I'm a messy programmer: in other words, I like to program straight on to the screen. Not for me the clean, aseptic approach of writing everything out beforehand.

Thus I often get to a point where I'm uncertain which variables I've used, which I haven't, and what values I've given those that I have put in. I'm sure that you, gentle reader, never do this, but just in case there's someone out there who does, DUMP could be exactly what he or she needs.

Family resemblance

The final two commands — AUTO and DELETE — have a family resemblance also. Their names explain virtually all. AUTO provides automatic line numbering, in whatever increments your heart desires, as you type furiously through your program-entering process. DELETE wipes whole blocks of lines out, between whatever numbers you specify.

That, at least must save wear and tear on the Return key but, ironically enough, the other commands available in Toolkit tend to lessen the usefulness of DELETE for me. I'd have loved it in those not-so-far-off days when I used subroutines tacked temporarily on to the front or rear of the main program.

And there it is: the Basic Programmer's Toolkit, with 2K of ROM firmware, consisting of a collection of machine language routines, all on one chip, adding nine (or 10, if you want to be finicky) new commands to your PET.

Is it worth £60? I would say yes.

Petsoft tell us:

- They are writing a British annexe to the Minimax manual with the intention of 'making a number of contributions aimed at the naive user.'
- This business software, available for the PET on Computhink mini-floppies, has been adapted and integrated for the Minimax. This means that programs from the suite can be run without having to 'bust through a series of complicated disc-to-disc steps.'
- Software titles which will be available early in the new year are Sales Accounting and Invoicing, Purchase Accounting, Stock Control, Word Processing and before the new financial year, Payroll.

Petsoft are also publishing a range of 50 utilities, languages and simple business routines. During the course of the year Petsoft will release some graphics packages supporting animated graphics, 3D plotting and a "small-talk" type package.

The CPU, in additiion to recognising all standard 6502 instructions, will also support 64 user programmable op codes. These can be used for Pascal and Forth operations.

CompuThink themselves have Pascal, a compiled Basic and Fortran under development for release early in 1980.

Conclusions

- **It's added a lot more fun to my programming**
- **It's made bugs much easier to find**
- **It's encouraged me to be a little more adventurous in trying out different methods, because I know, if they're not right, I'll be able to locate them and change them**
- **Maybe best of all, it has let me go back to half-finished programs from long ago, and tackle them again.**
- **Toolkit also makes conversion of old-ROM programs very much more simple.**
- **If you're any sort of 'hacker', what are you waiting for?**